Overview

This document covers the details of the benchmark applications associated with workstreams 1, 2 and 3 (see section C.3.4.1 and other RFP sections for definitions related to the concept of "workstreams").

A Microsoft Excel spreadsheet has been supplied for reporting initial results. Report all performance measures using only MPI (if run in parallel). Report only actual measurements in the original table. Copy the table to report performance values obtained using alternate technologies; clearly mark projected values and provide details as to how the projection was derived.

Benchmark Components

Components for Workstreams 1, 2 and 3

Summary:

Following the fundamental build/run design outlined in the head benchmark instruction document, the high level sequence for the benchmarks associated with workstreams 1, 2 and 3 is:

0) Obtain, build and test the Unidata netCDF v3.5 library. Note that the library must be built in a manner consistent with the intended build configuration for the models (e.g. consistent use of "64 bit" or "32 bit"). For the post processing benchmarks (pp<n>), you will also need to obtain the set of netCDF operators. See notes below on obtaining these items.

1) Download bench.base.tgz. Unpack the benchmark build directory bench using gunzip and tar –xvf.

2) Change directory to bench/etc and choose the mkmf.template.<architecture> make template for your architecture. The mkmf.template is included in the Makefile to provide certain make definitions such as the compiler invocation, compiler options, etc. You will probably have to modify the template to fit local conditions and current compiler, MPI and netCDF library requirements. You will have to create a new mkmf.template if the target platform is not included in bench/etc.

NOTE: <u>Default size of real must be 64; default size of integer may be either 32 or 64.</u>

3) Change directory to the target benchmark bench/<model>/exec

4) Edit the Makefile to define the BENCH variable for local directory structure and to include the desired mkmf.template.

5) Make the executable; record the time make required to compile and link the executable in the spreadsheet provided. Note that the Makefile contains a helpful functionality "localize" (i.e. make localize). This will copy all files used in the compilation to the local directory. Care must be taken in that source is compiled from its original location. Therefore modifications made to the "localized" source will have no effect on compilation.

6) Owing to their size, the benchmark run directories will be provided by tape. Offerors will supply a set of DLT format tapes to receive a copy of the benchmark run directories. Details for this process are pending and will be announced shortly.

The benchmark run directories associated with workstreams 1, 2 and 3 are:

- a) am2p13.esm.tgz (optional)
- b) mom4.esm.tgz (optional)
- c) cm2.ems.tgz (required)
- d) am2p13.hr.tgz (optional)
- e) mom4.hr.tgz (optional - not available with first release)
- f) cm2.hr.tgz (required - not available with first release)
- g) himf.vhr.tgz (required - not available with first release)
- h) pp<1-9>.tgz (required)

Unpack the benchmark model run directory using gunzip and tar –xvf.

7) Copy the executable to the run directory.

8) The model expects to read a file called input.nml. Multiple configuration dependent input.nml.<config> have been provided. Use UNIX ln or copy the desired input.nml.<config> to input.nml.

9) Run the executable using appropriate MPI invocation; pipe stdout to a text file. For example, a 30PE run on our Origin system would look like:

```
mpirun –np 30 fms_cm2.30.x |& tee out.30
```

Of course any other means of capturing stdout is also acceptable. This copy of stdout and the files dynam_integral.out and diag_integral.out must be returned with the benchmark results for the timing and the reproducibility test described below.

10) Report "Main loop" written to stdout by the program at the end of the run in the spreadsheet provided. Report value rounded to the nearest second.

11) Report per process memory used. The FMS infrastructure uses getrusage to obtain per process memory information. If your platform does not support getrusage, you will need to find another mechanism for reporting memory utilization. See bench/src/shared/memutils/memuse.c

12) Check results against verification files provided. The sum ordering required to make the exchange grid reproduce reduces performance and so timing should be done with the xgrid_nml variable make_exchange_reproduce set to false (applicable for am2p13, mom4 and cm2 benchmarks only, see the input.nml in each run directory). Given the highly non-linear nature of equations being solved, general consistency of output results is often the best that can be expected with the non-reproducing exchange grid. Verification output is located in bench/<model>/verification.

Model output files often have the same name regardless of PE count. Care must be taken not to accidentally overwrite output you wish to keep between consecutive model runs.

Note: The RFP benchmark will require a reproducibility test across PE count for some setting of the compiler, preferably that used for the performance benchmark itself. It is theoretically possible that high levels of compiler optimization might destroy reproducibility across PE count. To date, we have not seen this to be the case. Any mechanism causing loss of the capability to reproduce must be explained in detail. Use of such compiler optimizations will be weighed against the performance gain.

For each of the PE counts used as a benchmark data point, set make_exchange_reproduce to true in xgrid_nml and run the model for 8 simulation days. The bench/<model>/verification directory contains sample output as well as instructions for verifying reproducibility for the atmosphere and ocean components.

Note also that HIMF is not designed to be bitwise reproducible across PE count.

13) Benchmark timings from an SGI Origin 3000 cluster with 600MHz MIPS R12000 processors for each of the models may be found in bench/<model>/timings.Origin3000.<case>. Current experience on an SGI Altix cluster with 1.5GHZ Itanium processors indicates a performance increase of roughly 2x over the 600MHz MIPS processors. As one would expect, the scaling curve flattens more quickly than with the less capable, 600MHz MIPS Origin systems.

Background:

Parts of the NOAA Climate workstreams are run using the FMS (Flexible Modeling System) infrastructure. The components of FMS used in the benchmark include the FMS Bgrid atmospheric model, the FMS Modular Ocean Model v4 (MOM4) and an F90 implementation of the Hallberg Isopycnal Model (HIMF) now integrated within the FMS infrastructure. Note that the FMS infrastructure will require the availability of the Unidata netCDF library. Source code to build this library may be obtained from:

http://my.unidata.ucar.edu/content/software/netcdf/index.html

netCDF Version 3.5 is used for production work.

The netCDF operators are located at: http://nco.sourceforge.net/

Each of these libraries comes with self tests which are considered satisfactory to prove the function is working.

At this point in time, it is highly likely that the Climate portion of the benchmark will contain at least:

Item I: A global climate model consisting of an N45L24 (24 level), FMS Bgrid atmosphere coupled to an FMS MOM4 1-degree ocean. Designated as a prototype for an 'Earth System Model' (ESM), this component simulates a next generation low resolution climate with multiple tracers.

Item II: A global climate model consisting of an N90L40 (40 level) FMS Bgrid atmosphere coupled to an FMS MOM4, 1/3-degree ocean. This component simulates a next generation high resolution climate model.

Item III: An FMS HIMF, 1/6-degree hemispheric ocean model based on HIM (originally written in C, but now incorporated into the greater FMS infrastructure and re-written in F90). This component simulates a next generation very high resolution ocean model.

Item IV: A suite of tests derived from components of the coupled model post processing stream. These applications as implemented and run are single PE, generally high I/O.

Complete details for the formal benchmark will be forthcoming. However it is safe to assume that scalability studies of each of the benchmark models will be a component of the required benchmark data. In general it is desirable to run a model on the minimal set of processors required to supply sufficient real memory and report scaling data from the minimal set of processors through the number required to cause the performance curve to become substantially flat (i.e. "roll over"). Each item in the formal benchmark text will be defined with a minimum number of timing samples required to produce the scaling curve.

Performance data for each of these Items measured on the NOAA SGI Origin 3000 Cluster is provided.

Model Set Details

Item I is comprised of:

1) An N45L24 standalone atmosphere model w/ multiple tracers (grid is 144 x 90)
2) A 1 degree standalone ocean model w/ multiple tracers (grid is 360 x 200)
3) A global coupled climate model comprised of 1) and 2) designated as the N45L24 atm + 1deg ocn + atm and ocn tracers

Components 1) and 2) are supplied as an aid in porting 3) to a target platform. 1) and 2) will **not** be used as part of any benchmark by themselves and so **it is at vendor discretion that any resource be expended on the standalone components at all.**

Default size of real variables must be 64bit; default size of integer may be either 32bit or 64bit. More complete compiling and running instructions are included in following sections.

Item II is comprised of:

1) An N90L40 standalone atmosphere model (grid is 288 x 180)
2) A 1/3 degree standalone ocean model (grid is 1080 x 840)
3) A global coupled climate model comprised of 1) and 2) designated as the N90L40 atm + 1/3 deg ocn

Components 1) and 2) are supplied as an aid in porting 3) to a target platform. 1) and 2) will **not** be used as part of any benchmark by themselves and so **it is at vendor discretion that any resource be expended on the standalone components at all.**

Default size of real variables must be 64bit; default size of integer may be either 32bit or 64bit. More complete compiling and running instructions are included in following sections.

STATUS: Only the N90L40 standalone atmosphere is available at this time. The 1/3 degree ocean is in development and will be made available as soon as possible (likely early to mid September). The coupled model will follow ocean model release as soon as possible thereafter.

Item III is comprised of:

1) A 1/6 degree FMS HIMF standalone hemispheric ocean model and additional coarser resolution test models (benchmark grid is 2160 x 680)

"Smaller" (i.e. coarser grid) components are supplied as an aid in porting to a target platform. These smaller components will **<u>not</u>** be used as part of any benchmark by themselves and so **it is at vendor discretion that any resource be expended on the coarser resolution components at all.**

Default size of real variables must be 64bit; default size of integer may be either 32bit or 64bit. More complete compiling and running instructions are included in following sections.

Item IV is comprised of:

1) Post processing components: timavg, splitvars, plevel, ncrcat, ncap, ncatted, ncks and cpio

"Smaller" data components are supplied as an aid in porting to a target platform. These smaller components will **<u>not</u>** be used as part of any benchmark by themselves and so **it is at vendor discretion that any resource be expended on the smaller data components at all.**

Default size of real variables must be 64bit; default size of integer may be either 32bit or 64bit. More complete compiling and running instructions are included in following sections.

Status: The full benchmarks for Item IV will require large amounts of data. Tapes of the data in DLT format will be made available. Until that time, small test cases are provided. These smaller test cases will **<u>not</u>** be used as part of any benchmark by themselves and so **it is at vendor discretion that any resource be expended on the test cases at all.**

General Notes

This section is designed to give an overview of the items supplied with the benchmark. Details for compilation and model runs will be supplied with the benchmark directories.

<u>Building the executables</u>

The benchmark build directory structure has the following form:

bench/bin
bench/etc
bench/src.workstreams123
bench/<model>

where <model> is: am2p13, mom4, cm2 and pp. The am2p13 and mom4 are standalone models provided to aid porting; there is no requirement for the vendor to run these cases.

Each <model> directory contains the exec/ and src/ subdirectories. Note that src/ is simply a soft link to bench/src.workstream123. Thus care must be exercised since modification of src/ for one model may affect code used by another. The exec/ is the build directory and contains the Makefile for that model. Note that the bench/src directory contains a series of files named path_names.<model>. The path_names file contains a list of all source modules used by the model as well as the subdirectory path for that file. Among other things, path_names can be used to locate files of interest.

Verification output is located in bench/<model>/verification.

Makefiles have been provided for each benchmark application. The Offeror is free to modify the Makefile directly or rebuild the Makefile to suit local conditions using tools provided with this benchmark. The Makefile uses the “include” mechanism to import a platform specific template containing compiler option information. Sample templates may be found in the bench/etc directory.

The directory bench/bin contains the mkmf perl script used to create each Makefile. Additionally, the bench/<model>/scripts subdirectory contains a set of c-shell scripts which invoke mkmf to create the desired Makefile. While the mkmf default is to use but a single compile option line, it is possible to define compile options tailored to specific files. See the section on mkmf for specifics.

Some models such as the atmosphere can run with completely malleable executables (i.e. a single executable serves all PE configurations). Other codes such as the ocean models find significant performance improvements on some platforms when array components are fully specified at compile time through the STATIC_MEMORY preprocessing directive. One of the “costs” of the use of STATIC_MEMORY is that each PE configuration requires it’s own executable. MOM4 may be built in either fully malleable or STATIC_MEMORY mode; HIMF only has STATIC_MEMORY.

Note that the Makefile contains a helpful functionality “localize” (i.e. make localize). This will copy all files used in compilation to the local directory. Care must be taken in that source is compiled from it’s original location. Therefore modifications made to the “localized” source will have no affect on compilation.

NOTE: The FMS infrastructure uses the preprocessing “define” mechanism to create the numerous explicitly typed interfaces required by F90. As a result, there is constant redefinition of previously defined macro variables. Putting an explicit undef for each macro variable use prior to the next define is simply not practical or maintainable. Many preprocessor/compiler infrastructures emit warning messages when a previously defined macro variable is redefined. These messages become a nuisance and are suppressed on the current Origin platform. Unless the Offeror has explicit reason to believe that something is going wrong during preprocessing (such as failure to produce interfaces which cause the compilation to abort), then the messages should be suppressed or ignored.

Code optimizations

For the purposes of the benchmarks associated with workstreams 1, 2 and 3, the interpretation of "classes of changes for code optimization" are as follows:

Class A: Same as head instructions document – modifications to allow code to run to completion only.

Class B: modifications are source code changes to the parallel communication libraries. This includes use of communication libraries beyond MPI. Such changes are encouraged though maintenance of MPI will still be required for portability. The MPP library supporting workstreams 1-3 supports Coarray Fortran (CAF) and direct copy from remote global shared memory (GSM) as add-ons to it's MPI communication. See the section on MPP for details.

Class C: Same as head instructions document – modifications are limited to those which do not reduce code portability and which remain consistent with ANSI standard FORTRAN90/95 and C.

Class D: Same as head instructions document – those changes to application source not included in Classes A, B, or C. **Class D modifications are very strongly discouraged**.

All acceptable changes must produce output consistent with the verification provided as described with each benchmark.

As described in the instructions below, baseline performance numbers comprised of only Class A modifications will be required. MPP compiled with –Duse_libMPI will be the required "parallel framework" for this baseline where a communication library is required. Use of existing CAF or GSM within the MPI framework is permissible for the baseline in a manner consistent with Class A changes. Additional implementation of CAF or GSM within the FMS framework is considered a Class B change. MPI (or any other communication library) is clearly not applicable for systems which use compiler or operating system mediated AUTOMATIC parallelization for the baseline benchmark.

Running the model

Given the size of some of the input datasets, separate run directories have been provided as .tgz files (tar'd and gzipped). The Offeror will need to copy the executables built in the bench/ directories to the appropriate run directory. The run directories are named as the model targeted for that directory.

The AM2P13, MOM4 and CM2 models read an input.nml containing numerous runtime parameters in F90 namelist format. At the top of each input file are a couple of namelists the Offeror may wish to adjust. The "layout" parameters in ocean_domains_nml (models utilizing MOM4) and bgrid_core_driver (models utilizing AM2P13) control the number of PEs for each of the horizontal directions. The simulation time can be changed by adjusting the "months", "days", "hours", "minutes" and "seconds" parameters in coupler_nml. Adjusting the simulation time may prove useful for porting of the model. But the Offeror should make every effort to run the model for the full simulation length supplied with the benchmark.

Models utilizing AM2P13 must consider the "physics_window" parameter in the atmosphere_nml. The bgrid core maintains the capability to loop its operators over subdomains of the PE processing domain. For example see subroutines bgrid_physics_down and bgrid_physics_up in:

bench/src/atmos_bgrid/driver/coupled/bgrid_physics.f90

One of the main purposes of this feature is to control the size of automatic arrays in the operators. Further, the size of the subdomain will have performance implications for cache and/or vector length within the operators. The default setting is physics_window=0,1 which sends the operators a single j-slice at a time; physics_window=0,0 causes the entire domain to be operated on in a single call. The benchmarks are shipped with physics_window settings optimal for the Origin 3000. The Offeror may adjust these settings as necessary. It should be noted that setting the physics_window too large (e.g. 0,0) may cause runtime user stack overflow, especially for larger models on smaller PE counts. Given that error trapping for user stack memory is generally minimal, program behavior associated with stack overflow is notoriously bizarre. If one is fortunate, the program will dump core in the event of stack overflow.

The CM2 coupled models have an additional feature in that they may be run in either “serial” or “concurrent” mode. In this context, “serial” means that the model atmosphere, ice, land and ocean components run one after the other on the same set of processors. The “concurrent” mode allows the ocean model to run on one set of PEs while the remaining model components run on a different set of PEs. While the concurrent mode produces the best performance for current CM2 production runs, it does so through increased costs of the exchange grid. Thus, it is possible that serial mode will produce the best performance for some architectures. Examples of both concurrent and serial modes have been provided. Running in concurrent or serial mode is controlled by the “concurrent” parameter of coupler_nml.

The model infrastructure maintains a number of its own internal “stacks” and monitors for stack overflow. Most of these stack arrays are associated with exploiting “symmetric memory” on the current target architecture. This requires participating PEs to request memory at the same time during an F90 ALLOCATE call. These stacks work perfectly well without symmetric memory. But like any fixed stack, they can overflow and this will cause the model run to abort. While all tests provided have been run to completion, testing PE configurations with fewer processors may cause stack overflow. In the event of a stack overflow, the program should report which stack has suffered the problem and provide the size at which the current setting overflowed. New stack sizes can be provided through the fms_nml namelist parameter stated in the error message. As with all stacks, the stack cannot guess what’s coming past the point it failed. And so, it may take one or two tries to get the stack setting large enough. Thus, short initial runs are suggested. Overflows generally occur as the model grabs more memory for writing out data on exit.

All model input and output is either ascii or netCDF format. This should minimize I/O related porting issues. But it will require a functioning netCDF library.

All models except HIMF use the “exchange grid” to pass data between model components running on different grids (such as the atmosphere and ocean). The exchange grid can be run in two modes: “reproducing” and “non-reproducing”. The non-reproducing mode uses a PE count dependent methodology to perform certain sums as quickly as possible; the reproducing mode is PE count independent. In general, the non-reproducing mode is faster and so is used in production. The reproducing mode is for test purposes and as such, is useful in debugging problems. The Offeror should ensure the reproducing mode works for at least some set of compiler options, preferably all levels of optimization.

For the RFP response, the benchmark will require a reproducibility test across PE count for some setting of the compiler, preferably that used for the performance benchmark itself. It is theoretically possible that high levels of compiler optimization might destroy reproducibility across PE count. To date, we have not seen this to be the case. Any mechanism causing loss of the capability to reproduce must be explained in detail. Use of such compiler optimizations will be weighed against the performance gain.

For each of the PE counts used as a benchmark data point, set make_exchange_reproduce to true in xgrid_nml and run the model for 8 simulation days. The bench/<model>/verification directory contains sample output as well as instructions for verifying reproducibility for the atmosphere and ocean components.

Model output files often have the same name regardless of PE count. Care must be taken not to accidentally overwrite output you wish to keep between consecutive model runs.

Reporting initial results

A reporting template in the form of a Microsoft Excel spreadsheet is provided. In general, the Offeror will report the total time required for the Makefile to run from a clean directory using the UNIX time command. Further, it is highly desirable for the Offeror to report measurements from 6 or more PE counts for each of the benchmarks: CM2-ESM, CM2-HighRes and HIMF. As of the release date, only the CM2-ESM and atmosphere portion of CM2-HighRes are ready; the remaining portions will be released as quickly as possible. The PE counts should range from the lowest number of PEs on which the benchmark model can be run within an 8 hour period to the PE count past which the scaling curve is substantially flat.

Report value rounded to the nearest second.

At the Offeror's option, a second template may be used to report a projection of the initial timings to target platforms the Offeror may propose.

The MPP Parallel Communication Library

The MPP infrastructure comes with a couple of standalone tests. These tests may prove useful in general in porting the model components to the target platform. These tests are also useful when testing the use of enhanced communication features described below. See the Makefiles in bench/src/shared/mpp.

This section details advanced aspects of the parallel communication library used by workstreams 1-3. The features described are somewhat complex. Thus, the discussion is an introduction and designed to make the Offeror aware of these capabilities. Please contact the Government through official RFP channels with questions concerning implementation of these features.

For workstreams 1-3, parallel communication is supported by the MPP library (see bench/src/shared/mpp). Use of MPI within the MPP infrastructure is mediated through the –Duse_libMPI preprocessing option. All benchmark components ship with this preprocessing option enabled. Additional technologies are supported within the MPP – MPI implementation which affect (primarily) the "halo update" (mpp_update_domains), redistribution of data between different decompositions (mpp_redistribute) and gathering of local domains into the global field (mpp_global_field).

The first technology is called “call stacking”. This feature affects mpp_update_domains and mpp_redistribute and allows multiple calls to these routines to be made before forcing completion of the call set. The feature is enabled through use of the optional argument “complete” which takes logical values of “.true.” or “.false.”. During calls prior to the completion step (complete=.false.), input and output array address information is collected. For the last call in the sequence, complete=.true. and the actual communication routines are invoked. During this process, a structure is created containing information about the arrays involved, array dimensions, domain, etc. It is a requirement that all array participating in the “stack” have the same dimension and require the same type of update. For a sample implementation, see src/mom4/ocean_param/mixing/neutral/ocean_neutral_physics.F90, line 1695). This technology is detailed here because the implementation of this feature is incomplete within FMS at the time of this release. The Offeror may find it of value to implement this feature at specific sites within the code if communication bottlenecks are identified.

Use of direct read from remote global shared memory (GSM) is supported as an additional compile time option within the MPI infrastructure. GSM is invoked using the –Duse_GSM preprocessing option in addition to –Duse_libMPI. When used, GSM bypasses the pack/send/recv/unpack communication model of MPI in favor of direct copy from (for example) a domain residing on a remote PE to the halo of the copying PE. This remote read requires strict and possibly costly process synchronization. To amortize the cost of synchronization, the “stacking” feature described above is generally required to increase performance.

Coarray Fortran (CAF) is also supported as an additional compile time option within the MPI infrastructure. CAF is invoked using the –Duse_CAF preprocessing option. The implementation is similar in concept, if more elegant than the GSM implementation. CAF pointers are used to read remote memory after proper synchronization.

Note that even when compiled with the additional option (GSM or CAF), the option is not used by default within the application. Rather, an optional argument must be added to the targeted MPP function to cause the enhanced version’s invocation. This allows for incremental introduction of the capability as well as backward compatibility. The mechanism for utilizing the enhanced feature when compiled is the “complete=.true.” optional argument for mpp_update_domains and mpp_redistribute and “new=.true.” for mpp_global_field. GSM and CAF are coupled to the “complete” argument for the updates since both technologies require explicit process synchronization across the communicating PE set. Thus, call stacking would is likely to be required to see performance enhancement. On the other hand, there is nothing preventing their use for an individual call; simply set complete=.true. for that call.

To date, only the MOM4 model has added call stacking in released code. Offerors are free to add call stacking, CAF or GSM to any portion of the FMS infrastructure. Such changes are considered class B and so performance obtained with such changes should be reported as an addition to the baseline performance. As implemented, the call stacking feature relies on a persistent incoming array address for lookup of the associated communication data structure (recreating the structure can be expensive, especially when using GSM since remote address information needs to be passed between PEs). Thus, call stacking and GSM are not designed for use with automatic arrays. This problem can be circumvented by use of another optional argument which allows the calling routine to hang on to a handle for the communication structure, bypassing the lookup. Since the address information itself is gathered locally each time the sequence is called, general call stacking used with MPI and CAF would continue to work. GSM will fail automatic arrays as the implementation requires a reading PE to have the remote address explicitly and that information is exchanged only the first time the sequence is called.

As noted, details for these additional technologies are rather complicated. Offerors are encouraged to contact the Government with questions should they wish to use GSM, CAF or additional call stacking.

Internal timing for the FMS infrastructure on the Origin platform is handled through src/shared/mpp/nsclock.c. On platforms other than Origin, mpi_wtime is used as the timer; nsclock.c has an ifdef which should prevent compilation on non-SGI platforms. The Offeror is free to replace mpi_wtime with any other timer of sufficient resolution. Since only high level loop timings are required, however, mpi_wtime is likely to be sufficient.

MKMF

The perl script mkmf in bench/bin was used to create the Makefiles for workstreams 1-3. Offerors are free to modify the Makefiles directly if this is the most convenient method for introducing compile options tailored to individual modules (i.e. options beyond those which can be specified by the single FFLAGS variable in the mkmf.template located in bench/etc).

Use of the C shell scripts mk_<model> in bench/<model>/scripts allows a new Makefile to be generated in the bench/<model>/exec directory. Note that the process overwrites any existing Makefile in the exec subdirectory. Further, use of mkmf within the mk_ scripts references a "path_names.<model>" file in bench/src. The path_names file contains a list of source files. Use of mkmf in conjunction with the path_names file allows one to generate a Makefile with file specific compile lines.

To regenerate a Makefile, change to the bench/<model>/scripts subdirectory, edit the script for local conditions and invoke mk_<model>. The mk_ scripts will generate a new Makefile and initiate the compilation. Note that while the Makefiles provided use **relative** directory paths making the bench directory and its workstream 1-3 components "portable" with respect to compilation, Makefiles generated by mkmf use **absolute** paths. To cause the generation of a file specific compile line, one modifies the file entry in the associated path_names file. For instance, the MOM4 file ocean_neutral_physics.F90 gets compiled as:

```
$(FC) $(CPPDEFS) $(CPPFLAGS) $(FFLAGS) -c -I$(BENCH)/mom4/mom4.90/src/mom4/ocean_core
$(BENCH)/mom4/mom4.90/src/mom4/ocean_param/mixing/neutral/ocean_neutral_physics.F90
```

To alter this default for the MOM4 or CM2 models, modify the path_names.mom4 or path_names.cm2 file as in the path_names.mom4.special_compiler_option with:

```
$(FC) $(CPPDEFS) $(CPPFLAGS) $(MY_FFLAGS) –c
```

placed on the line referencing ocean_neutral_physics.F90 where MY_FFLAGS is defined in the mkmf.template. Note that one must specify the full Makefile line since the override process has no way (for example) to associate MY_FFLAGS as a replacement for a particular option such as FFLAGS.

While the tools are a little complicated to get used to, they can be a tremendous aid when one has to generate multiple Makefiles. The Offeror should contact the Government through official RFP channels with questions should they wish to use with the mkmf tool.